A Modular Peripheral to Support Self-Reconfiguration in SoCs

Andrés Otero, Ángel Morales-Cas, Jorge Portilla, Eduardo de la Torre, and Teresa Riesgo
Centro de Electrónica Industrial, Universidad Politécnica de Madrid
{joseandres.otero, jorge.portilla, eduardo.delatorre, teresa.riesgo}@upm.es, angel.morales.cas@alumnos.upm.es

Abstract— **In this paper, a solution to support the run-time readback, relocation and replication of cores in embedded systems with dynamic and partial reconfiguration capabilities is presented. The proposal shows a peripheral structure that allows an easy integration and communication with the rest of the system, including an API to make the reconfiguration details to be more transparent to software applications. Differently to other proposals, all functionality is implemented in hardware, achieving a higher reconfiguration speed. In addition, different design decisions have been taken in order to increase the portability of the solution to existing and, possibly, future FPGAs. Finally, a use case is provided, which shows the features of this module applied to the run-time scaling of a hardware coprocessor.**

Keywords-component; FPGA; Dynamic Reconfiguration; core relocation; Scalability

I. INTRODUCTION

The emergence of commercial FPGAs with dynamic and partial reconfiguration (DPR) features has opened promising research opportunities in the field of reconfigurable computing. The benefits and possibilities offered by this technique have been extensively reported, mainly in academic works [1] [2] [3] [4]. An important one is that partial reconfiguration reduces time and memory overheads compared with full reconfiguration. In addition, it allows the design of specific hardware for each particular application, providing advantages, both regarding power consumption and performance. Furthermore, dynamic partial reconfiguration allows the implementation of complex systems in constrained devices, and enhances run-time adaptability for embedded systems working under variable environments. Consequently, due to these advantages, and the limits of silicon integration, reconfigurable computing is seen like a must for future embedded systems.

Despite these advantages, the main obstacles to provide partially reconfigurable solutions with commercial purposes are the lack of commercial design flows and tools supporting it [5], as well as the need of having specific knowledge on dynamic reconfiguration techniques. In addition, the particularity of each device, and the changes from one generation of reconfigurable devices to the next one, makes difficult to create portable and upgradable solutions. To deal with dynamic reconfiguration from scratch, different issues have to be solved. Some of them are the creation of partial configuration files to reconfigure only the desired portion of the device, the tools to relocate these modules in any arbitrary position, or the control of the configuration port and the communication infrastructure compatible with the DPR technique [6].

Regarding the configuration ports, different interfaces are available. Xilinx FPGAs contain the ICAP, an internal port that allows the partial reconfiguration of the device. This leads to the concept of self-reconfigurable embedded system, a system that can modify its functionality autonomously at run-time. To control the ICAP, Xilinx provides a wrapper called HWICAP that can be integrated in the system like a peripheral through the IBM PLB CoreConnect bus [7].

Regarding the generation of partial bitstreams, that is, configuration files to reconfigure only the desired resources of the device, traditionally there has always been an important gap between the silicon availability and the apparition of design tools and flows supporting this feature. In addition, partial reconfiguration files are addressed to the position of the device where they were initially mapped. As a result, either a different partial bitstream is generated in advance for each possible reconfigurable region, or the final position of each block is restricted to the original one. With the second option, the system can enter into stall states, if it requires the execution of a certain hardware block, and the corresponding reconfigurable region is not available. An alternative, with low computing overhead and bitstream storage requirements, is to modify the target position of the hardware modules at run-time, during the reconfiguration process. Some solutions exist to carry it out, including both software and hardware implementations, like those reported in [8] and [8]. However, hardware approaches have restricted functionality and are very device dependent, while software ones provide lower reconfiguration rates, because of their high overhead.

The work described in this paper provides an easy and fast hardware based solution for the readback, relocation and replication of partial bitstreams. The main purpose of this solution is to provide digital systems designers the possibility of making logical designs that take advantage of the partial dynamic reconfiguration, without having a previous preparation in this field. Additionally, the hardware system was designed to be easily ported to new devices and families of Xilinx FPGAs. This module is designed as a peripheral module, including integration with the rest of the system by means of the CoreConnect technology, and a software API to

increase independence between the family-specific hardware implementation and the software applications using the dynamic reconfiguration capabilities offered by the peripheral. This independence will be proved with the adaptation of this solution not only across devices within a family, but also between different families. Also, the reconfiguration control process provided by the block, may allow future online modifications (also called mutations) of the bitstream.

The rest of the paper is organized as follows. In section II, a brief description of the Xilinx Virtex FPGAs reconfiguration details is provided. Section III includes an overview of the state-of-the art of the existing hardware solutions for bitstream manipulation, while in section IV the approach of this paper is introduced. In sections V and VI, implementation details, both at hardware and software levels are described, and finally, in section VII, implementation results are given.

II. XILINX VIRTEX RECONFIGURATION DETAILS

Before introducing the details of the solution proposed in this paper, it is necessary to explain some concepts about the Xilinx Virtex architecture and its bitstream structure.

Partial reconfiguration of Xilinx FPGAs is based on the modification of the content of the SRAM that controls the logic configuration of each element of the device. The smaller addressable unit of this configuration memory is called a frame, and defines the minimum reconfiguration granularity. Frames for Virtex families prior to Virtex-4 are arranged in columns that span the device entirely from top to bottom. On the other side, Virtex-4, Virtex-5 and Virtex-6 devices have more complex architectures, since they are composed by stacked rows of configurable blocks, called clock regions. Each clock region is composed by columns of configurable elements. Each frame expands the height of a row/clock region. As a result, not the full height of the device has to be reconfigured at each time, permitting a 2D reconfiguration scenario. This can be seen in Figure 1.

Access to the internal configuration memory may be done, among other possibilities, through the internal registers of the ICAP port. Partial reconfiguration implies to provide some configuration commands, the register programming for those registers in the area being reconfigured, and the bitstream data itself. For the purpose of this paper, the registers of interest are the FAR, where the address of the frame to be accessed has to be stored; the CRC, a checksum of the configuration data; the COR, where the configuration options are selected; the FDRI, where the FAR data to be configured is stored, and the FDRO, that provides readback configuration information. In addition, a command register (CMD) is used to trigger the internal state machine of the ICAP. Details of this process can be found in the configuration user guides like [10] [11].

To reconfigure a region, the FAR values corresponding to this region have to be generated. In Virtex devices, the FAR is created like the composition of different fields, as it can be seen in Figure 2. The first one is the position of the frame, in the top or in the bottom of the device. Then, the clock region where it is situated has to be indicated.

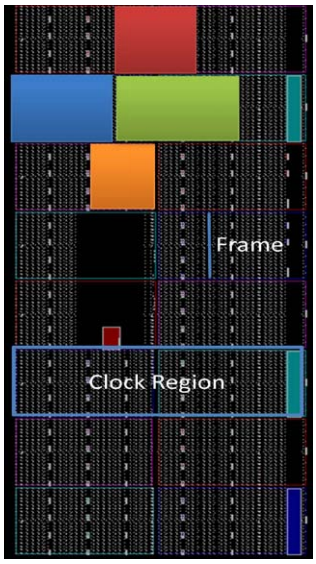

Figure 1. Two dimensional partial reconfiguration in Virtex-5

Afterwards, the Major Address, that is, the column inside the clock region has to be selected. Finally, the Minor Address, the position of the frame inside a column is identified. The number of frames needed to configure a column depends on the kind of logical elements that it contains. Other field, called block type, selects the configuration layer of the device. In this work, only type 0 elements will be addressed, that is, the *interconnect and block configuration* type. The concrete value of all these parameters, as well as the number of resources of the FPGA and their relative positions, depend on the particular device.

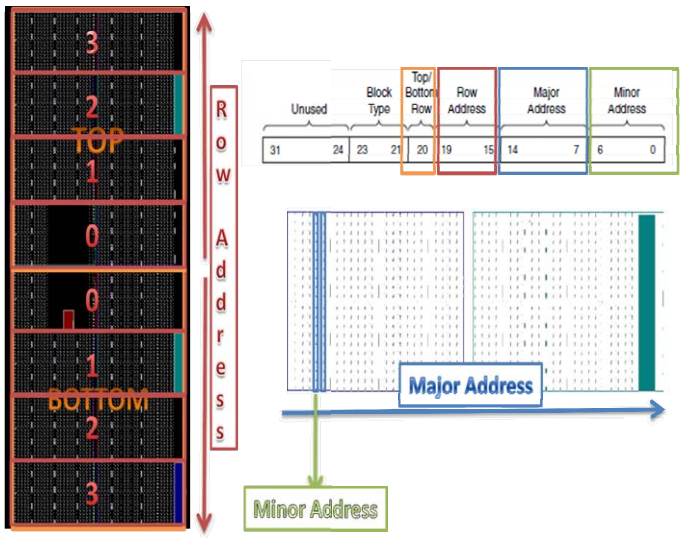

Figure 2. Virtex-5 FAR format

As it has been already said, the solution provided by Xilinx to create self-reconfigurable platforms is called HWICAP. This is a wrapper of the ICAP primitive, including some internal memories, an ICAP control state-machine and the ICAP instantiation. This hardware system has a peripheral structure,

to be integrated on a SoC, including some Peripheral Interface Blocks to communicate with the PLB, as well as a software driver. This driver includes functions to access the FPGA resources, but not to reallocate arbitrary reconfigurable blocks, neither to perform the readback of an arbitrary region.

III. STATE OF THE ART

Since dynamic reconfiguration of commercial FPGAs became feasible, the relocation of presynthesized bitstreams has been addressed in different works. A first classification among the existing solutions can be done according to the location of the bitstream manipulation, that is, off-chip or on-chip. The first external solution was PARBIT [12]. However, in this work, the objective of the designed peripheral is to provide an embedded system with enhanced dynamic reconfiguration capabilities. Consequently, only on-chip solutions will be analyzed. Among them, a further classification can be carried out depending on the processing approach, existing both embedded software and hardware solutions. Examples of software-based solutions are XPART, a C version derived from the set of JBits Java classes and pBITPOS[13], an embedded version of BITPOS. A good summary of software-based solutions is provided in [14]. However, since software based solutions offer too high overhead for run-time operation, this state-of–the art will be restricted to hardware approaches.

Focusing on embedded hardware solutions, a remarkable approach is the REPLICA (Relocation per online Configuration Alteration) filter proposed in [15]. This solution is based on the concept of the bitstream filtering during relocation, in order to reduce the process overhead. Basically, the idea is to implement a finite state machine that parses the bitstream during the configuration, identifying the different fields and commands. Among these fields, it is necessary to detect the FAR, in order to replace the original addresses where the block was synthesized, with the final addresses where it will be relocated, as well as the CRC. The generation of the new addresses where each frame has to be relocated is the heart of the filter. Since this approach is restricted to Xilinx Virtex and Virtex-E FPGAs, it only addresses 1D relocations. The address calculation is reduced to obtain a constant offset value to be added to the original Major address. A correction factor has to be included to skip BRAM columns. The calculus of the FAR in Virtex-E devices is more difficult, so some necessary operations are performed in advance. The filter also includes an FPGA Type Decoder, which from the device identification code generates some specific parameters of the chip, necessary to calculate the addresses. In addition, a block to generate the new CRC value, after bitstream manipulation, is included. The overhead of this block is null, since the next MJA is calculated while a column is being reconfigured. The output of this block is the manipulated bitstream that can be used to feed the ICAP, or other interfaces like the SelectMAP. Details of the control of the reconfiguration port are out of the scope of this work. Authors also provide a configuration manager to control the data transfers from the bitstream memory to the port. The relocator can be situated between the bitstream memory and the configuration manager. A new version including support for Virtex-2/-Pro devices, called REPLICA2Pro, was introduced in [8]. It also includes support for the relocation of tasks that make use of BlockRAM and multiplier columns.

The BIRF (Bitstream Relocator Filter), presented in [16], is similar to REPLICA in the sense that it is also based on a filter approach, and is restricted to 1D relocation in Virtex FPGAs. However, BIRF claims to have a better performance, but no performance information was included in REPLICA. The main difference is a simplification of the Parser FSM, limiting its behavior to detect the FAR and the CRC. An enhanced version of this proposal, reported in [8], solves the bidimensional relocation problem, addressing Virtex-4 and Virtex-5 devices. The FAR is also calculated by the parser, and it depends on the differences between virtex-4 and virtex-5 devices. An optimized version is provided, including a bypass value for the CRC that avoids the calculation of the new CRC.

Another approach is the Self-reconfiguring Platform (SRP) by Blodget et al. in [17] and [18], which includes the idea of attaching the hardware reconfiguration subsystem as a peripheral of the embedded MicroBlaze processor, using the CoreConnect Bus technology. Inside the hardware subsystem, a cache is included, with capacity to store a single configuration frame. The software component of the platform defines a low-level API, in charge of the ICAP-cache memory control, and a high level API programmed in C language called XPART, derived from JBits. As a result, it contains methods to random access bitstreams, as well as to relocate partial bitstreams. This software component contains the read/modify/write operation for configuration data, which enables fine-grain hardware modification, like changing LUT equations, as well as the copyCLBModule function, that copies a rectangular region of configuration memory, and pastes it in another position. This can be very useful to some applications, avoiding the access to the external memory. In [19], based on this Self-Reconfigurable Platform, the concept of merge reconfiguration is introduced. The process followed is to, instead of directly writing the new configuration data in the device memory, the former one is previously read, and frame by frame merged with new data. This allows the inclusion of static routing, as well as the reconfiguration of blocks occupying less than a frame height. This idea was initially addressed to virtex-2 devices, but has also been successfully tested in Virtex-4. In [20], a software version of this read-modify-write method is also exploited.

The PRR-PRR reconfiguration approach introduced in [21], provides frame relocating from a partial reconfigurable region (PRR) in the device, into another region. This is done in a frame by frame basis, in order to accelerate the relocation process and to reduce the bitstream storing requirements. However, this approach has a huge overhead of repeating the different sequences of the header of each pair of frames, having no possibility of reconfiguring blocks from the external memory. Furthermore, the algorithm to relocate a full rectangular region is implemented in software.

The solution proposed in this paper is incremental regarding previous works, but contributing with some original implementation ideas, and enhanced functionality.

IV. PROPOSED RELOCATION SOLUTION

The solution for the bitstream relocation proposed in this paper can be considered like a hardware-based one, with enhanced functionality. As a result, this platform offers fast configuration, with features offered by software solutions, but still with reduced area overhead, and following a generic design approach to increase the portability of the solution. Furthermore, it is compatible with cascaded bitstream filters in order to produce mutations of the bitstream for other purposes like, but not limited to, evolvable hardware solutions, redundancy addition for fault tolerance or side-channel attack protection.

The hardware system is integrated as a peripheral structure, to be used with the microprocessor embedded in the FPGA (hard or soft core), making it compatible with the CoreConnect Technology. In addition, its file structure fits perfectly with Xilinx's EDK peripheral cores, so a user with no expertise in dynamic reconfiguration, can just add it to an EDK project the same way any other core is added. The system designer will have access to the resources provided by the hardware block, by means of a software API.

In addition of the bitstream relocation, among the hardware implemented options, bitstream readback, replication and relocation of full configurable columns have been implemented. This approach also allows performing fine-grain modifications during the relocation process. As a result, this peripheral can be seen like a final solution to easily provide dynamic reconfiguration capabilities developed so far. Another characteristic that can be very useful to certain applications is the copy and paste property, from a position in the internal memory, to a different one. This can be very useful in applications like the scalable coprocessors shown in [22]. This approach also allows a read-modify-write operation, like the proposed in [19].

The peripheral module has been designed following a generic and modular design approach, in order to easily adapt the core with new functionality, to other bus protocols, as well as to other FPGA families. Also, to improve the portability to new devices and between different members of the same family, the module in charge of address generation relies on a VHDL package that describes the elements of the FPGA that can be changed and resynthesized. Furthermore, an important structural difference, compared with other approaches, is that this solution, instead of relocating partial bitstreams, self-generates headers and tails of the bitstreams internally, and only receives the purely configuration data as an input. As a result, the platform has been designed so that it can be adapted to other families of FPGAs just by exchanging specific VHDL packages. In addition, the size of the configuration data of the core is reduced; on the other side, the relocation process is easier, since no parser has to be implemented. Furthermore, to allow a faster portability to new devices, as well as to simplify the generation of the ICAP control signals, the Xilinx HWICAP is included like a subcomponent. This ICAP already includes the configuration control machine. In the following sections, hardware implementation details, the software interface and the implementation results will be shown.

V. IMPLEMENTATION DETAILS

As has been already said, the peripheral module is completely compatible with the CoreConnect technology, allowing the easy integration of this block with the rest of the system. So, an IPIF has been included, as well as some control and data registers, accessible from the processor, and a FIFO memory. This memory stores the configuration data from the processor, as well as readback configuration frames, to be replicated and relocated in different regions of the reconfigurable device.

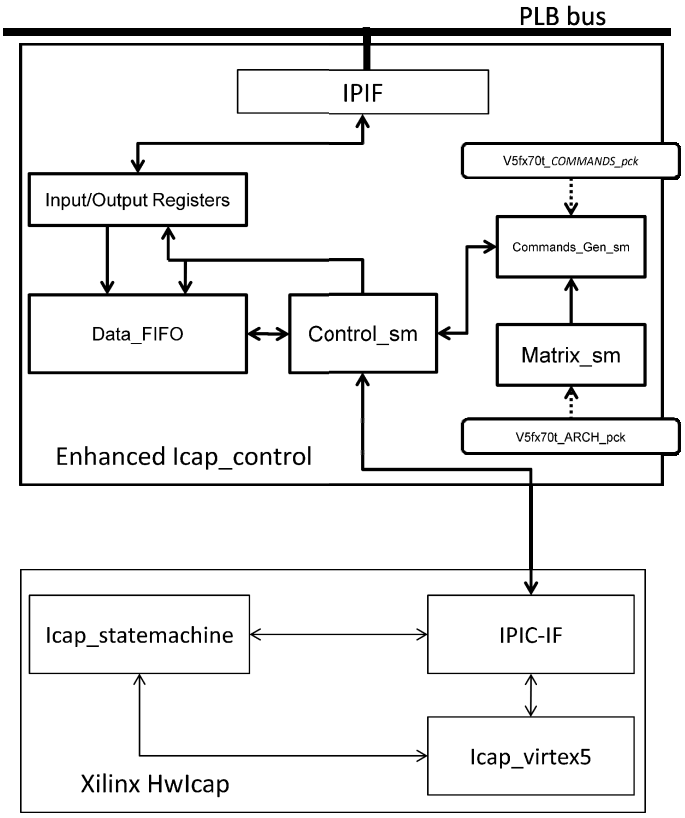

Figure 3. Dynamic Reconfiguration Peripheral Hardware Architecture

In addition, the hardware system includes some blocks in charge of control issues, the generation of the FAR, the generation of the header and the tail, as well as the control of the ICAP. In the following subsections, further details of the hardware implementation are provided.

A. VHDL packages

Reconfiguration procedures for all devices of the same Virtex family are equivalent. However, they differ in a lot of parameters, specific values, both architectural dependent, and also regarding the state-machine control details. This affects both the generation of the necessary commands to perform the partial reconfiguration, but also the generation of the FAR addresses. To solve this problem, instead of storing all the parameters for all devices in the family, two VHDL packages

are generated. The first one includes the specific architecture details to generate the FAR, and another one with the command sequences that constitute both header and tail of the bitstream. The elements of the FPGA architecture are indexed in XY coordinates and represent FPGA columns, not frames. When the peripheral is instantiated and resynthesized for a different device, the parameters stored in these files are used by the tool, creating specific hardware for each particular device. For new devices in the market, these packages might be created easily, whenever the structure is kept similar. This way, it will be possible to shorten the development of reconfigurable systems when new families appear in the market, thus shortening the tools gap before mentioned.

B. FAR Calculation

The inputs of this module are the XY coordinates of the lower-left and upper-right corners of the rectangular area that are going to be read or written. The block takes the information stored in the architecture device package to generate every frame address that span the rectangular region that will be reconfigured. This subsystem is a simple state machine that just reads the proper elements of the architectural package. It goes through each element of the package's array and takes its frame address fields. In order to generate the minor addresses, it increments it from zero to the number of frames of the column selected at that moment. When all the addresses of the selected column are generated, it goes to the next column. This process goes on until the last column is reached. As a result, new FPGAs with different architectures could be easily reconfigured.

At the output of this block, a FIFO memory is used, to store the generated addresses. This FIFO allows the synchronization with the data coming from the processor. When all addresses have been generated and sent, a false address (an address in which every bit is set to '1') is sent to the FIFO to indicate that the previous address was the last one.

C. Bitstream Commands generation

The idea of this block is very similar to the FAR generator. It reads the sequence of commands from the commands vhdl package, and feeds the ICAP with it. In addition, it includes the generated FAR, for each frame, in the proper position of the bitstream.

D. ICAP wrapper

The differences in the control and the interface of the ICAP primitive, depending on the specific device, make difficult the generation of portable solutions. Therefore, instead of instantiating the ICAP port directly in this solution, the HWICAP wrapper provided by Xilinx is used. The connection with the wrapper is done through the standard IPIF included in the HWICAP. This IPIF is completely standard, and compatible with new generations or versions of this wrapper.

E. Control blocks

Other modules of the system are small state-machines in charge of the generation of different control signals, like the control of the ICAP through the IPIF, as well as the execution of some tasks like the inclusion of headers and tails, or the information merging from different sources (data from external sources or from readback, relocated addresses, headers and tails).

VI. SOFTWARE API

The proposed peripheral includes a software API to ease its integration in the system, as well as to hide the reconfiguration details to software applications using it. The proposed interface is based on the definition of a set of commands that are stored in the peripheral registers, allowing the control of the hardware subsystem from the processor.

The block has six Registers: the commands one, a data register, and four coordinates' registers to communicate two positions in the FPGA. Possible content of the commands register is shown in Table 1.

Table 1. Software API commands

Command	Effect
READ2EXTERNAL	Reads the configuration of a region from the FPGA configuration memory and sends it to the processor.
READ2INTERNAL	Reads the configuration of a region from the FPGA configuration memory and stores it in the internal FIFO
WRITEFROMEXTERNAL	Writes the configuration of a region from the processor to the FPGA configuration memory.
WRITEFROMINTERNAL	Writes the configuration of a region from the peripheral FIFO memory to the configuration memory.

In addition, two bits of this register are used to indicate data transferences between the processor and the peripheral. The addresses of the rectangular regions involved in the operations are introduced through the registers X0, XF, Y0, and YF. These addresses indicate conventional coordinates of the FPGA, beginning from the bottom left corner.

VII. EXPERIMENTAL RESULTS

In this section, implementation results, as well as a use case of the reconfiguration peripheral, are provided.

A. Implementation results

Synthesis values are compared with the similar approaches. The first analysis is offered in terms of resource occupation, as Table 2 shows. The values of this table have been selected trying to compare the results of each work in similar conditions or operation profiles. Consequently, the REPLICA2Pro results correspond to a profile without CRC checking, while data corresponding to BIRF2D area has been estimated from relative percentage results, and HWICAP is removed in all cases.

Table 2. Resource occupation results

Approach	Slices	Memory Requirements
Proposal (Without HWICAP)	296	4 blocks of 36KBbits
BIRF 2D	80	No internal memory
REPLICA2Pro	322	No internal memory
HWICAP	594	No Block RAM is used

According to this table, values obtained for the proposed module are comparable with Replica2Pro, in spite of the implementation of the new functionalities in hardware, and the portable generic design approach. The introduction of the enhanced control capabilities does not imply a significant area overhead, since the total used slices are, for instance, 2% of the total resources of a Virtex-5 FX70T, which is in the small-medium range of the family. In this version, four 36 Kbits memory blocks have been dedicated to store partial bitstreams, but this value can be adapted to different applications.

Regarding the maximum operation frequency, the proposed solution can operate at up to 208 MHz. The reported frequencies of the reference works in similar conditions, that is, using the synthesis and simulation information, are 160 MHz for BIRF (also on Virtex-5) and 35 MHz for Replica2D (on Virtex-4). In spite of these values, the running frequency when control blocks are integrated in the system is limited also by the maximum operation rate of the ICAP. Xilinx guarantees a maximum frequency of 100MHz. Consequently, at this speed, the addition of relocation features does not include an extra overhead. However, recent works have proved the feasibility of ICAP overclocking [22], so this maximum speed is very important for the performance of future implementations of the reconfiguration block.

The behavior of the block can be seen in the oscilloscope captures shown in figures 4 to 7, corresponding to different steps and details of the reconfiguration process.

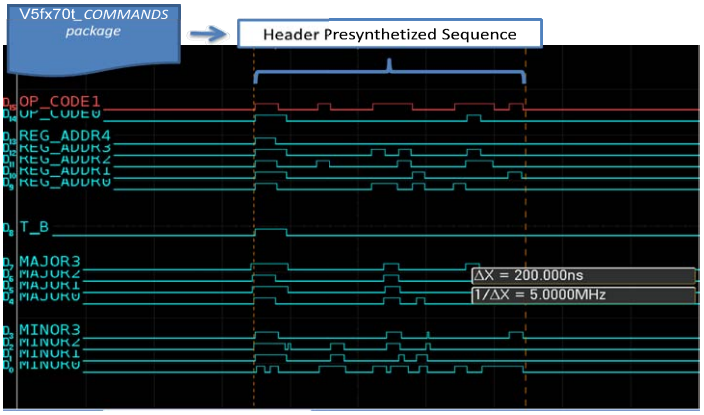

Figure 4. Header sequence generated to initiate a writeback process from the vhdl package

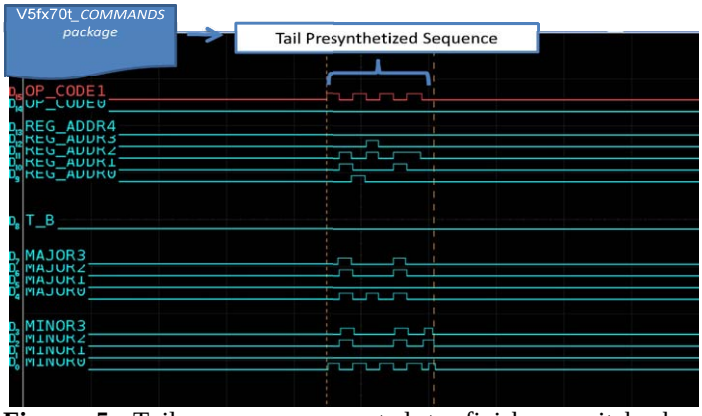

Figure 5. Tail sequence generated to finish a writeback process from the vhdl package

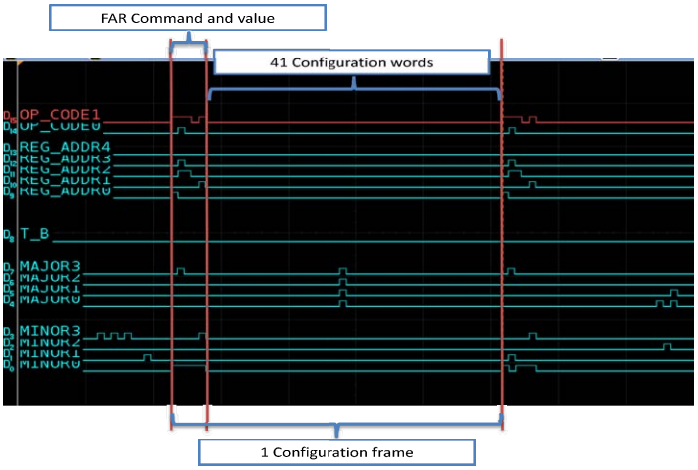

Figure 6. Sequence for the reconfiguration of a frame, including the FAR command and its value, and the 41-words of a configuration data frame (for the target Virtex-5)

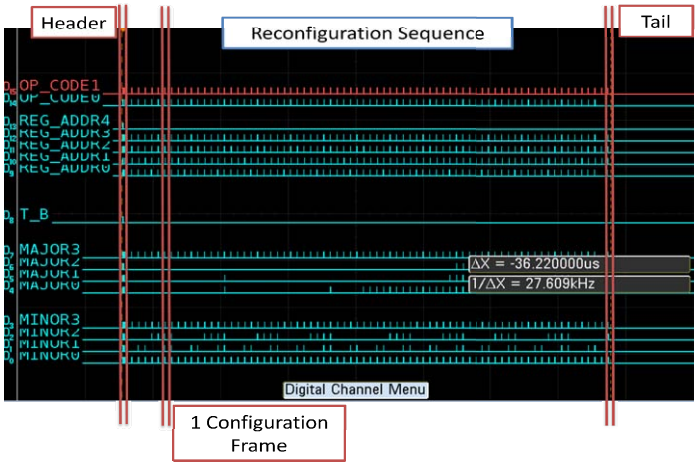

Figure 7. Reconfiguration sequence including the header, the succession of configuration frames and the final tail.

In Figure 4, the header sequence introduced before a reconfiguration process is shown, while the tail is reported in Figure 5. To reconfigure each frame, as can be seen in Figure 6, it is necessary to introduce in the ICAP the FAR command and the generated address value, followed by the corresponding configuration data. To reconfigure an arbitrary region, the process of a single frame has to be sequentially repeated, preceded by the header, and finishing with the tail. The full reconfiguration process is seen in the image Figure 7. The reconfiguration time of a region can be calculated as follows (in clock cycles):

$$Cycles_{Readback} = ReadHeader_{Length} + NumberofFrames \times \left(FAR_{Length} + frame\_length\right) + ReadTail_{Length}$$

$$Cycles_{Writeback} = WriteHeader_{Length} + NumberofFrames \times \left(FAR_{Length} + frame\_length\right) + WriteTail_{Length}$$

In Table 3, actual values for these parameters are shown.

Table 3. Actual number of clock cycles of each field of the reconfiguration sequence for Virtex-5 FX70T

Field	Readback	Writeback
Header	6	14
FAR	5	5
Tail	2	9 + 1 Pad Frame = 50

B. Use case application

To validate the block, the proposed peripheral module has been used to create a dynamically scalable coprocessor. Scalable coprocessors are a possibility to adapt the functionality of some hardware tasks, depending on run-time variable applications requirements, or also on changing system conditions, like available power or chip area. The solution proposed in this paper, is also seen like a possibility to carry out fast size adaptation of scalable coprocessors. To prove the correctness of this solution, it has been employed together with the architecture proposed in [22]. The idea is to reconfigure the peripheral from a 3 × 3, to 5 × 5 size. However, due to the homogeneity of the modules, a single processing element of the array can be read back, and later replicated in all the new positions. As it can be seen in the Figure 8, the element has to be replicated into 16 different positions. Each processing element requires 72 reconfiguration frames (2 CLB columns by 36 frames for a CLB column). According with the previously reported equations, the readback process takes 100 cycles, and the write process 1536. At 100 MHz, the required time for this operation is 16,36 us.

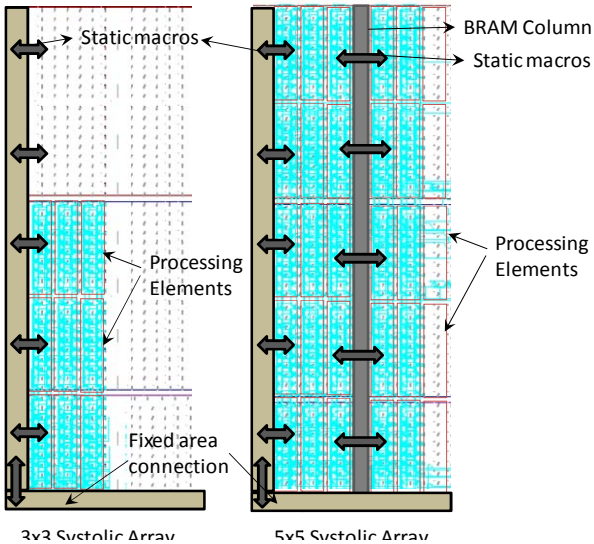

Figure 8. Run-time scalable coprocessor

.

In this case, since the readback data is already in the peripheral memory, no overhead for the transference from the external memory is incurred. Furthermore, to generate new processing elements, the proposal of this paper can also be used, configured like a readback to the external memory. For larger reconfiguration areas which are read from external sources, the provision of data to this block requires fast data access to the peripheral, but [23] shows that maximum speed may be achieved by using DMA schemes.

VIII. CONCLUSIONS AND FUTURE WORK

Real time applications that make use of run-time dynamic reconfiguration are becoming a key design topic, because they combine the speed of hardware solutions with the flexibility of reconfigurable systems. In this context, and specially when dealing with restricted devices (in both logic resources and/or power), fast and autonomous reconfiguration are of main importance.

The proposed solution provides extended features, comparable with software based approaches, for run-time reconfiguration

modules, but at speeds that are close to the maximum ones, given today's reconfiguration port achievable speeds. Additionally, the block is seamlessly integrated with the bus technologies available for SoPC systems, and the abstraction level at which reconfiguration is handled alleviates the system designer to have deep knowledge of reconfiguration skills (taking apart the knowledge required to produce a reconfigurable block layout, which is assumed to be done by a hardware designer, with much higher expertise in reconfiguration).

An effort towards generalization in the way different FPGA families behave with dynamic reconfiguration has been done, and the design of the peripheral module based on VHDL configuration packages opens up an opportunity to rapidly migrate the reconfiguration process to new coming families, with reduced development times. Experimental results show that a full featured hardware based solution still requires very little area overhead, at very high speeds.

Future work goes in three directions. On one side, the possibility of adding more bitstream filters (mutators) opens up the enhancement of the block with other features like increased fault tolerance by automatic circuit modifications, protection against side channel attacks, etc. Second, the 300 MHz frontier reported in [23] for maximum programming speed is a challenge for our all-at-one relocation and programming feature. Third, the use of this module into a variety of scenarios where fast reconfiguration is required, like video processing with inter-frame algorithm adaptation, for instance, or evolvable systems, are broader research lines that are being considered presently.

IX. Conclusions and Future Work

This work was supported by the Artemis program under the project SMART (Secure, Mobile Visual Sensor Networks Architecture) with number ARTEMIS-2008-100032.

References

[1] Todman, T.J.; Constantinides, G.A.; Wilton, S.J.E.; Mencer, O.; Luk, W.; Cheung, P.Y.K.; , "Reconfigurable computing: architectures and design methods," *Computers and Digital Techniques, IEE Proceedings* - , vol.152, no.2, pp. 193- 207, Mar 2005

[2] Wang Lie; Wu Feng-yan; , "Dynamic Partial Reconfiguration in FPGAs," *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on* , vol.2, no., pp.445-448, 21-22 Nov. 2009

[3] Wu, K.; Madsen, J.; , "Run-time dynamic reconfiguration: a reality check based on FPGA architectures from Xilinx," *NORCHIP Conference, 2005. 23rd* , vol., no., pp. 192- 195, 21-22 Nov. 2005

[4] Paulsson, K.; Hubner, M.; Becker, J.; , "Exploitation of dynamic and partial hardware reconfiguration for on-line power/performance optimization," *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on* , vol., no., pp.699-700, 8-10 Sept. 2008

[5] Santambrogio, M.D.; , "From Reconfigurable Architectures to Self-Adaptive Autonomic Systems," *Computational Science and Engineering, 2009. CSE '09. International Conference on* , vol.2, no., pp.926-931, 29-31 Aug. 2009

[6] Hagemeyer, J.; Keltelhoit, B.; Koester, M.; Pomnann, M.; , "A Design Methodology for Communication Infrastructures on Partially Reconfigurable FPGAs," *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on* , vol., no., pp.331-338, 27-29 Aug. 2007

[7] http://www.xilinx.com/ipcenter/processor_central/coreconnect/coreconnect.htm

[8] Kalte, H. and Porrmann, M. 2006. REPLICA2Pro: task relocation by bitstream manipulation in virtex-II/Pro FPGAs. In *Proceedings of the 3rd Conference on Computing Frontiers* (Ischia, Italy, May 03 - 05, 2006). CF '06. ACM, New York, NY, 403-412. DOI=

[9] Corbetta, S.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D.; Spoletini, P.; , "Internal and External Bitstream Relocation for Partial Dynamic Reconfiguration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.17, no.11, pp.1650-1654, Nov. 2009

[10] http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

[11] http://www.xilinx.com/support/documentation/user_guides/ug360.pdf

[12] Horta, E.L.; Lockwood, J.W.; Taylor, D.E.; Parlour, D.; , "Dynamic hardware plugins in an FPGA with partial run-time reconfiguration," *Design Automation Conference, 2002. Proceedings. 39th* , vol., no., pp. 343- 348, 2002

[13] Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; Joly, D.; , "Virtex II FPGA Bitstream Manipulation: Application to Reconfiguration Control Systems," *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on* , vol., no., pp.1-4, 28-30 Aug. 2006

[14] Filho, F.V.; Horta, E.L.; , "Development Tools for Partial Reconfigurable Systems," *Programmable Logic, 2008 4th Southern Conference on* , vol., no., pp.249-252, 26-28 March 2008

[15] Kalte, H.; Lee, G.; Porrmann, M.; Ruckert, U.; , "REPLICA: A Bitstream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems," *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* , vol., no., pp. 151b- 151b, 04-08 April 2005

[16] Ferrandi, F.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D.; , "Dynamic Reconfiguration: Core Relocation via Partial Bitstreams Filtering with Minimal Overhead," *International Symposium on System-on-Chip, 2006.*, vol., no., pp.1-4, 13-16 Nov. 2006

[17] Blodget, B., James-Roxby, P., Keller, E., McMillan, S., and Sundararajan, P. : “A self-reconfiguring platform”, In Proceedings of the Field-Programmable Logic and Applications, September 2003. Springer-Verlag

[18] Blodget, B.; McMillan, S.; Lysaght, P.; , "A lightweight approach for embedded reconfiguration of FPGAs," *Design, Automation and Test in Europe Conference and Exhibition, 2003* , vol., no., pp. 399- 400, 2003

[19] Sedcole, P.; Blodget, B.; Becker, T.; Anderson, J.; Lysaght, P.; , "Modular dynamic reconfiguration in Virtex FPGAs," *Computers and Digital Techniques, IEE Proceedings* - , vol.153, no.3, pp. 157- 164, 2 May 2006

[20] Hiibner, M.; Schuck, C.; Kiihnle, M.; Becker, J.; , "New 2-dimensional partial dynamic reconfiguration techniques for real-time adaptive microelectronic circuits," *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on* , vol.00, no., pp.6 pp., 2-3 March 2006

[21] Sudarsanam, A.; Kallam, R.; Dasu, A.; , "PRR-PRR Dynamic Relocation," *Computer Architecture Letters* , vol.8, no.2, pp.44-47, Feb. 2009

[22] Otero, A.;Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; “Generic Systolic Array for Run-Time Scalable Cores”, *6th International Symposium on Applied Reconfigurable Computing, ARC* 2010, Bangkok, Thailand, March 17-19, 2010

[23] C. Claus, R. Ahmed, F. Altenried, W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems", *6th International Symposium on Applied Reconfigurable Computing, ARC* 2010, Bangkok, Thailand, March 17-19, 2010